



**Министерство науки и высшего образования Российской  
Федерации Федеральное государственное бюджетное  
образовательное учреждение  
высшего образования  
«Московский государственный технический  
университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)**

**ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»**

**КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И  
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО  
ДИСЦИПЛИНЕ:**

**ТИПЫ И СТРУКТУРЫ ДАННЫХ**

**ТЕМА: ОБРАБОТКА РАЗРЕЖЕННЫХ МАТРИЦ**

**Студент Поздышев А. В.**

**Группа ИУ7-31Б**

**Название предприятия НУК ИУ МГТУ им. Н. Э. Баумана**

**Студент \_\_\_\_\_ Поздышев А. В.**

**Преподаватель \_\_\_\_\_ Барышникова М. Ю.**

**2024**

<b>ОГЛАВЛЕНИЕ</b>	
<b>ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ .....</b>	<b>3</b>
<b>ОПИСАНИЕ ВНУТРЕННИХ СТРУКТУР ДАННЫХ.....</b>	<b>4</b>
<b>ФУНКЦИИ ПРОГРАММЫ .....</b>	<b>5</b>
<b>ОПИСАНИЕ АЛГОРИТМА.....</b>	<b>8</b>
<b>НАБОР ТЕСТОВ .....</b>	<b>9</b>
<b>МЕТОДИКА ЗАМЕРОВ.....</b>	<b>11</b>
<b>ВЫВОД .....</b>	<b>12</b>
<b>КОНТРОЛЬНЫЕ ВОПРОСЫ .....</b>	<b>12</b>

## ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ

Реализация алгоритмов обработки разреженных матриц, сравнение эффективности применения этих алгоритмов со стандартными алгоритмами обработки матриц при различном размере матриц и степени их разреженности.

1. Смоделировать операцию умножения вектора-строки и матрицы, хранящихся в указанной форме, с получением результата в форме хранения вектора.
2. Произвести операцию умножения, применяя стандартный алгоритм работы с матрицами.
3. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

### Исходные данные

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов (CSC): - вектор A содержит значения ненулевых элементов; - вектор IA содержит номера строк для элементов вектора A; - вектор JA, в элементе  $N_k$  которого находится номер компонент в A и IA, с которых начинается описание столбца  $N_k$  матрицы A. Вектор-строка хранится в 2х объектах: - вектор B, содержащий значения ненулевых элементов - вектор JB, параллельный вектору B, содержащий индексы ненулевых элементов

### Способ обращения к программе

Запуск программы осуществляется через консоль: ./app.exe. После чего необходимо ввести путь к файлу, содержащему матрицу. В результате пользователю будет представлено меню опций.

### Меню программы:

1. Вывести матрицу на экран.
2. Вывести csc матрицу на экран.
3. Ввести матрицу.
4. Ввести вектор-строку.
5. Вывести вектор-строку.
6. Умножить CSC матрицу на вектор-строку.
7. Умножить матрицу на вектор-строку (Стандартный алгоритм).
8. Считать матрицу из файла.
9. Считать вектор-строку из файла.
10. Перевести матрицу в CSC формат.

11. Перевести CSC матрицу в стандартную.
12. Сравнить методы умножения матриц на вектор-строку.
0. Выход.

Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Некорректный ввод: Пустое поле ввода (ожидание ввода пользователя);
- Некорректный ввод: В матрице символы;
- Некорректный ввод: Размеры матрицы отрицательны;
- Некорректный ввод: Размеры матрицы превышают максимальные;
- Некорректный ввод: Размер вектора превышает максимальный;
- Некорректный ввод: Символы вместо размеров матрицы;
- Некорректный ввод: В векторе индексы повторяются;
- Некорректный ввод: Вектор-строка только из нулей;
- Ошибка выделения памяти: Размер матрицы не помещается в кучу.

## ОПИСАНИЕ ВНУТРЕННИХ СТРУКТУР ДАННЫХ

Разреженная матрица хранится в формате CSC в sparseMatrix\_t

```
typedef struct sparse_matrix // Разреженная матрица.  
{  
    int *A;  
    int *IA;  
    int *JA;  
    int numNonZero;  
    int row_len;  
    int num_rows;  
} sparseMatrix_t;
```

1. int \*A – указатель на массив значений.
2. int \*IA – указатель на массив индексов, соответствующих значениям.
3. int \*JA – указатель на массив, содержащий описание столбцов элементов A и IA.
4. int numNonZero – кол-во ненулевых элементов.
5. int row\_len – длина строк.

6. int num\_rows – кол-во строк.

Вектор-строка хранится в структуре vec\_t

```
typedef struct vector // Структура для вектора строки.  
{  
    int *B;  
    int *JB;  
    int numNonZero;  
    int vec_len;  
} vec_t;
```

1. int \*B – указатель на массив значений вектора.
2. int \*JB – указатель на массив индексов, соответствующих значениям.
3. int numNonZero – кол-во ненулевых элементов.
4. int vec\_len – длина вектора.

Обычная матрица размером nxm представлена в виде двух блоков. В первом блоке содержатся указатели на строки второго блока, в котором хранятся значения матрицы. Структура int \*\*matrix.

## ФУНКЦИИ ПРОГРАММЫ

Create\_matrix

Заголовок:

```
int **create_matrix(size_t const n, size_t const m);
```

Функция создает матрицу с размером n на m. Функция принимает размеры матрицы n и m. Возвращает указатель на матрицу.

Create\_sparseMatrix

Заголовок:

```
sparseMatrix_t *create_sparseMatrix(int **matrix, int num_rows, int row_len);
```

Функция создает разреженную матрицу в формате CSC. Функция принимает указатель на матрицу matrix, размеры матрицы num\_rows и row\_len. Возвращает указатель на sparseMatrix\_t.

## Create\_vector

Заголовок:

```
vec_t *create_vector(int *values, int *indexes, int none_zero_count, int vec_len);
```

Функция создает вектор-строку. Функция принимает указатели на массив значений `values`, массив индексов `indexes`, кол-во ненулевых элементов `none_zero_count`, длину вектора `vec_len`. Возвращает указатель на `vec_t`.

## Multivec\_csc

Заголовок:

```
int *multivec_csc(size_t *res_len, sparseMatrix_t *matrix, vec_t *vector);
```

Функция осуществляет умножение разреженной матрицы на вектор-строку. Функция принимает указатели на размер получаемого вектора `res_len`, указатель на разреженную матрицу `matrix`, указатель на вектор-строку `vector`. Возвращает указатель на полученный вектор.

## Multivec\_mat

Заголовок:

```
int *multivec_mat(size_t *res_len, int **matrix, size_t const n, size_t const m, int *vector, size_t const n_v);
```

Функция осуществляет умножение матрицы на вектор. Функция принимает указатели на размер получаемого вектора `res_len`, указатель на матрицу `matrix`, ее размеры `n` и `m`, указатель на вектор `vector` и его размер `n_v`. Возвращает указатель на полученный вектор.

## Csc\_convert\_tomat

Заголовок:

```
int **csc_convert_tomat(size_t *n, size_t *m, sparseMatrix_t *csc);
```

Функция осуществляет перевод матрицы формата CSC в стандартную. Функция принимает указатели на размеры получаемой матрицы `n`, `m`, указатель на матрицу `csc`. Возвращает указатель на полученную матрицу.

## Mat\_convert\_tocsc

Заголовок:

```
sparseMatrix_t *matconvert_tocsc(int **matrix, size_t const n, size_t const m);
```

Функция осуществляет перевод матрицы в формат CSC. Функция принимает указатели на матрицу `matrix`, ее размеры `n`, `m`. Возвращает указатель на полученную матрицу формата CSC.

`Print_vector`

Заголовок:

```
void print_vector(vec_t *vector);
```

Функция осуществляет вывод вектора-строки. Функция принимает указатели на сам вектор `vector`.

`Print_csc`

Заголовок:

```
void print_csc(sparseMatrix_t *csc);
```

Функция осуществляет вывод матрицы в формате CSC. Функция принимает указатель на саму матрицу `csc`.

`Print_matrix`

Заголовок:

```
void print_matrix(int **matrix, size_t const n, size_t const m);
```

Функция осуществляет вывод матрицы в стандартном виде. Функция принимает указатель на саму матрицу `matrix` и ее размеры `n`, `m`.

`Input_vector`

Заголовок:

```
int input_vector(vec_t **vector);
```

Функция позволяет ввести вектор-строку с клавиатуры. Функция принимает двойной указатель на сам вектор `vector`. Возвращает код ошибки.

`Input_matrix`

Заголовок:

```
int input_matrix(int **matrix, size_t const n, size_t const m);
```

Функция позволяет ввести матрицу с клавиатуры. Функция принимает двойной указатель на матрицу и ее размеры. Возвращает код ошибки.

`Read_matrix`

Заголовок:

```
int read_matrix(FILE *file, int **matrix, size_t const n, size_t const m);
```

Функция позволяет считать матрицу из файла. Функция принимает файловую переменную `file`, двойной указатель на матрицу и ее размеры. Возвращает код ошибки.

`Fget_vector`

Заголовок:

```
int fget_vector(char *filename, vec_t **vector);
```

Функция позволяет считать вектор-строку из файла. Функция принимает имя файла `filename`, двойной указатель на вектор `vector`. Возвращает код ошибки.

## ОПИСАНИЕ АЛГОРИТМА

Алгоритм считывает данные из файла, затем выдает пользователю выбор действий.

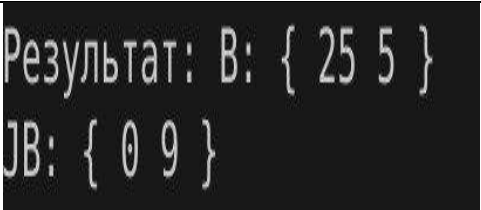
1. Действие 1:
  - a. Вывод матрицы на экран.
2. Действие 2:
  - a. Вывод матрицы в формате CSC.
3. Действие 3:
  - a. Осуществляет ввод матрицы.
  - b. Матрица обновляется.
4. Действие 4:
  - a. Осуществляет ввод вектора-строки.
  - b. Вектор-строка обновляется.
5. Действие 5:
  - a. Вывод вектора-строки.
6. Действие 6:
  - a. Проверка состояния матрицы CSC и вектора-строки.
  - b. Умножение матрицы CSC на вектор-строку.
  - c. Сохранение времени выполнения операции.
  - d. Вывод полученного вектора.
7. Действие 7:
  - a. Проверка состояния матрицы и вектора-строки.
  - b. Перевод вектора-строки в стандартный формат.
  - c. Умножение матрицы на вектор.



- d. Сохранение времени выполнения операции.
- e. Вывод полученного вектора.
- 8. Действие 8:
  - a. Освобождение матрицы.
  - b. Считывание матрицы из файла.
  - c. Обновление матрицы.
- 9. Действие 9:
  - a. Освобождение вектора-строки.
  - b. Считывание вектора-строки из файла.
- 10. Действие 10:
  - a. Проверка состояния матрицы.
  - b. Перевод матрицы в формат CSC
  - c. Обновление CSC матрицы.
- 11. Действие 11:
  - a. Проверка состояния матрицы CSC
  - b. Перевод в стандартный формат.
  - c. Вывод матрицы.
- 12. Действие 12:
  - a. Проверка на наличие замеров.
  - b. Вывод последних замеров.
- 13. Действие 0:
  - a. Выход из программы.

## НАБОР ТЕСТОВ

Тест	Входные данные	Выходные данные
Некорректный ввод: размеры матрицы отрицательные	10 -10	ERR_IO: Ошибка некорректного ввода
Некорректный ввод: размеры матрицы - символы	A 10	ERR_IO: Ошибка некорректного ввода
Некорректный ввод: В матрице символ	10 10 1 2 a 2 1 2 4	ERR_IO: Ошибка некорректного ввода

Некорректный ввод Кол-во ненулевых эл. больше размера вектора.	10 11	ERR_IO: Ошибка некорректного ввода
Некорректный ввод вектора: индекс предыдущего больше	10 5 1 0 1 5 1 3	ERR_IO: Ошибка некорректного ввода
Некорректный ввод вектора: отрицательный индекс.	10 5 1 0 1 5 1 -7	ERR_IO: Ошибка некорректного ввода
Некорректный ввод вектора: отриц. кол-во ненулевых элементов	10 -1	ERR_IO: Ошибка некорректного ввода
Ошибка выделения: размер файла не помещается в кучу.	<файл>	ERR_ALLOC: Ошибка выделения памяти.
Корректный ввод матрицы и вектора	Матрица размером 10x10, вектор длины 10	

Корректный ввод матрицы и вектора	Матрица размером 100x100, вектор длины 100	Результат: B: { 15 17 14 17 11 14 21 13 16 12 8 14 6 15 18 17 11 6 5 14 13 5 11 17 19 9 9 10 25 19 7 15 12 0 0 14 3 11 2 15 23 10 24 7 4 17 14 12 16 2 10 10 26 9 7 12 23 10 14 15 12 14 2 4 4 14 10 5 12 10 9 12 27 20 22 2 9 10 12 8 15 9 8 17 4 10 23 11 16 6 4 27 6 3 15 16 22 15 } C: { 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 }
-----------------------------------	--	---

## МЕТОДИКА ЗАМЕРОВ

Вычисление замеров сортировок проводилось следующим образом: для каждого вида операции для опр. % 0 находилось среднее время за 10 замеров. Матрица размером 1000x1000.

## ЗАМЕРЫ

% 0 в матрице	Стандартный алгоритм, мс.	Алгоритм для CSC и вектора строки, мс.
10	1.96	2.51
20	1.98	2.16
30	1.96	1.9
40	1.98	1.5
50	1.98	1.4
60	1.98	1.09
70	1.96	0.81
80	1.96	0.57
90	1.98	0.32
100	1.98	0

При 30% нулей в матрице метод для разреженной матрицы в формате CSC начинает превосходить по времени стандартный алгоритм.

## ПАМЯТЬ

% 0 в матрице	Размер матрицы, байты	Размер CSC матрицы, байты
10	4008000	7204044
20	4008000	6404044
30	4008000	5604044
40	4008000	4804044
50	4008000	4004044
60	4008000	3204044
70	4008000	2404044
80	4008000	1604044
90	4008000	804044
100	4008000	0

Хранение в CSC формате для матрицы размером 1000x1000 при 50% нулевых элементов объемы памяти сравниваются. При увеличении кол-ва нулевых элементов хранение становится все более эффективным

## ВЫВОД

Для операций над разреженными матрицами лучше использовать такие схемы хранения как: CSR, CSC и др. С уменьшением кол-ва нулевых элементов время выполнения операции умножения матрицы на вектор уменьшается и – наоборот. В 30% нулевых элементов время выполнения алгоритмов сравнивается, а при более чем в 50% уменьшается более чем 1,5 раза.

При увеличении кол-ва нулевых элементов размер стандартной матрицы остается неизменным, пока размер матрицы в формате CSC быстро уменьшается. Так при хранении в CSC формате для матрицы размером 1000x1000 рост эффективности в хранении начинается с 50% нулевых элементов.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?

Разреженная матрица — это матрица, в которой значительная часть элементов равна нулю.

Существуют такие виды как: CSR, CSC и хранение в формате индексов.

CSR – сжатое хранение строк.

CSC – сжатое хранение столбцов.

2.Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?

Под хранение разреженной матрицы выделяется память под три вектора, в случае CSC – это A, IA, JA. Размеры двух первых определяются кол-вом ненулевых элементов, а для последнего – кол-вом столбцов. Для стандартной матрицы память выделяется пол блок из  $n \times m$  элементов, где  $n$  и  $m$  размеры матрицы.

3.Каков принцип обработки разреженной матрицы?

Обработка разреженной матрицы включает в себя различные методы и алгоритмы, которые позволяют эффективно работать с матрицами, содержащими много нулевых значений. Основная цель — снизить потребление памяти и улучшить производительность при выполнении математических операций.

4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?

Из замеров видно, что эффективней всего использовать стандартные операции, если в матрице малое кол-во нулевых элементов. Уже при 30% нулевых элементов в матрице алгоритм CSC обгоняет по эффективности времени выполнения стандартный и значительно уменьшает объемы выделяемой памяти.